Knowledge-Based Resource Allocation for Collaborative Simulation Development in a Multi-tenant Cloud Computing Environment

Gongzhuang Peng, Hongwei Wang, Jietao Dong, and Heming Zhang

Abstract—Cloud computing technologies have enabled a new paradigm for advanced product development powered by the provision and subscription of computational services in a multi-tenant distributed simulation environment. The description of computational resources and their optimal allocation among tenants with different requirements holds the key to implementing effective software systems for such a paradigm. To address this issue, a systematic framework for monitoring, analyzing and improving system performance is proposed in this research. Specifically, a radial basis function neural network is established to transform simulation tasks with abstract descriptions into specific resource requirements in terms of their quantities and qualities. Additionally, a novel mathematical model is constructed to represent the complex resource allocation process in a multi-tenant computing environment by considering priority-based tenant satisfaction, total computational cost and multi-level load balance. To achieve optimal resource allocation, an improved multi-objective genetic algorithm is proposed based on the elitist archive and the *K*-means approaches. As demonstrated in a case study, the proposed framework and methods can effectively support the cloud simulation paradigm and efficiently meet tenants' computational requirements in a distributed environment.

Index Terms—Cloud computing, Collaborative simulation, Resource scheduling, Knowledge-based engineering, Multi-objective optimization, Radial basis function neural network (RBFNN)

1 Introduction

THE rapid developments and applications of information technologies have brought a revolutionary change to modern product development [1]. Through the use of high-performance computers, powerful software packages and efficient network services, the design and development process is accelerated and product quality is improved at the same time. As design work is done more often by geographically and temporally distributed design teams involving different roles such as modelers, domain experts, validation and verification experts and end users of various backgrounds, complex product development becomes increasingly collaborative and integrated. Thus, the need of developing collaborative working platforms for multiple users to share heterogeneous resources and conduct design tasks in a collaborative and distributed environment has been raised.

Service-Oriented Architectures (SOA) has emerged as a solution to this problem by utilizing services as the fundamental elements for developing applications. In SOA, all kinds of computational resources are encapsulated as services and delivered to customers according to their personal requirements on a pay-per-use pricing basis [3]. Collaborative intelligent manufacturing has been increasingly adopted under the SOA paradigm [4]. Boeing, for example, is using SOA principles behind a new PaaS platform called the Boeing Edge which promises to reshape the way Boeing connects with its customers in the airline business [5]. Among various SOA frameworks, Cloud computing has become very popular owing to that it enables customers to acquire applications, platforms, and a wide variety of resources from third parties that are distributed all over the world [2]. By moving to the cloud model, modern product development advances towards collaboration, intelligence, digitalization and sustainability. The architecture of a multi-tenant distributed simulation system is shown in Fig.1, which includes four main parts together with the service providers and tenants working in a cloud computing environment.

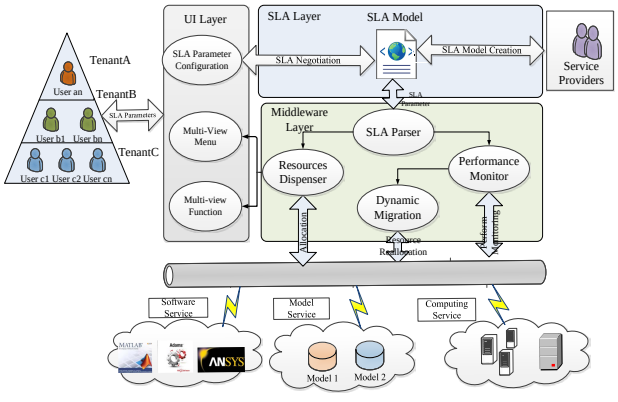


Fig. 1. Architecture of the system

Computational resources are traded between the service providers and the consumers or tenants. These different roles may come from different enterprises, different departments within an enterprise or different individuals within a department. As the development of complex

- *G. Peng, J. Dong and H. Zhang are with the National CIMS Engineering Research Center, Department of Automation, Tsinghua University, China. E-mail: pgz12@mails.tsinghua.edu.cn*
- *H. Wang is with the School of Engineering, University of Portsmouth, Portsmouth PO1 3DJ, UK*

products involves the collaboration of different people from different domains in completing a series of tasks, an effective and efficient mechanism is necessary to analyze the tasks and then identify and match resource requirements in terms of their quantities and qualities. Most of the analysis work is now done based on the experience obtained in previous projects. By reusing this kind of implicit knowledge, a Radial Basis Function Neural Network (RFBNN) is established to estimate the resources required by each tenant.

In a cloud computing environment, the qualified services are usually numerous and dynamic, which makes it a challenging task to allocate resources. The total cost is a widely-used criterion to measure the performance of allocation schemes [6]. As Green Manufacturing (GM) and Sustainable Development (SD) are playing even more important roles in the business and industry of the 21st century [10], [11], the energy consumption of a cloud center has to be taken into consideration as well [12]. Therefore, load balance is considered as one of the optimization indexes during the resource allocating process. As some problems such as services security need to be addressed urgently, the priority of each tenant also becomes important to the resource allocating process, and as such a priority-based satisfaction index is proposed to ensure the tenants' highest satisfaction. Thus, the performance of resource allocation in cloud computing is evaluated using a Multi-Objective Optimization (MOO) problem with three indexes in this research.

Compared with the single objective optimization (SOO) problems, the MOO problems are characterized by incommensurability and contradiction between different objectives. Specifically, incommensurability refers to the fact that there is no uniform measure for various objectives while contradiction means performance of one objective is improved at the cost of making others worse. A common method for analyzing such a problem is to transform a MOO problem into a SOO problem using the method of weighting aggregation. However, the optimization result is largely determined by the weights of different factors and determination of these weights is very difficult without comprehensive knowledge of the problem. Researchers have made significant progress in overcoming this difficulty by finding out all the non-dominated solutions which are called the set of Pareto front solutions.

As the solution space of resource scheduling in cloud computing is too huge to explore for the conventional operational research (OR) algorithms, some artificial intelligence methods have been proposed in recent years, such as the Genetic Algorithm (GA), the Particle Swarm Optimization (PSO), the Simulated Annealing (SA), the Ant Colony Optimization (ACO) [33-35] and so on. Owing to the simplicity of operation and the power of effect of the GA approach [14], it has been widely applied to timetabling and scheduling problems [13]. Nevertheless, genetic algorithms have some inherent limitations. For example, they have a tendency to converge towards local optima and an exponential increase of search space and time will be incurred when they are applied to large-scale problems. To overcome these limitations for solving the problem of cloud computing resources scheduling effectively and efficiently, the K-means based GA (KGA) has been proposed by taking advantages of the traditional GAs and local search.

The main contributions of this paper include:

1) A radial basis function neural networks is established as a knowledge model for estimating the required resources of each tenant.
2) A novel resource scheduling model is formulated for a highly heterogeneous cloud environment by considering the tenant priority, load balance and energy consumption criteria.
3) An improved KGA algorithm is developed based on the elitist strategy and the *k*-means approach and its effectiveness and feasibility is demonstrated in a resource scheduling case study.

The rest of this paper is organized as follows. Section 2 reviews some previous studies related to this work. Section 3 explains in detail the process of resource allocation for cloud simulation and introduces the QoS interpreting model based on RBFNN. Section 4 describes the multi-objective problem in resource scheduling and details the formulation of the mathematical model. Section 5 describes the improved KGA algorithm and the performance indicators. After that, implementation of a prototype system and the evaluation of the algorithms proposed are given in Section 6. Finally conclusions and future work are discussed in Section 7.

2 Related Work

Cloud computing has been applied in a wide range of domains such as collaborative simulation, advanced manufacturing [21], medical engineering [17] and social science [22]. Most of the research on cloud computing was about platform development [23], virtualization approaches and resource provision algorithms. Among these categories, the third one is exactly the focus of this paper.

Since modern product development becomes increasingly knowledge-intensive, much research work has been done on capturing and reusing process knowledge [19]. Yang *et al.* developed a UML-based profile to capture expert's knowledge for automating the instantiation of a computing environment [18]. Kwang *et al.* took advantage of semantic knowledge and proposed three types of reasoning to search services that match consumers' functional and technical requirements including similarity reasoning, compatibility reasoning, and numerical reasoning [15]. Ramachandran *et al.* used a tenant requirements model (TRM) and a tenant provider model (TPM) to represent tenants' requirements and providers along with their different attributes and behaviors [37]. RBFNN has been utilized in numerous fields for engineering, social science and so on ascribed to its excellent learning capability [20].

The objectives of resource scheduling are diverse according to different service providers and task types. Chaisiri *et al.* proposed a stochastic programming model

to reduce the total resource provisioning cost. Wei *et al.* took both optimization and fairness into account in maximizing computational resource's utility and minimizing expenses [28]. Wu *et al.* proposed a customer driven SLA-based resource provision algorithm to minimize cost by minimizing resource and penalty cost as well as to improve customer satisfaction level (CSL) by minimizing SLA violations [2]. Tao *et al.* considered energy consumption as one of the optimization indexes in their case library based hybrid genetic algorithm [12]. Chen *et al.* designed a Resource Matching Algorithm (RMA) which focused on the resource refusal rate, resource average waiting cycle and resource utilization based on a resource virtualization model [36].

In a multi-tenant environment, priorities of service requests from different tenants should be taken into consideration, as some problems such as services security issues and administrator requesting processes need to be urgently addressed. Thus, the priority-based tenant satisfaction measure is proposed in this research in the construction of the multi-objectives model in addition to the total cost and multi-level load balancing measures.

A particular category of solutions for this scheduling problem is to represent user requests with a graph and apply methods based on the graph theory to reduce the overall complexity. Yu *et al.* developed a *k-shortest* path algorithm to minimize bandwidth consumption in a virtual network [25]. Chrysa *et al.* modeled the networked cloud request as a weighted undirected graph and solved the optimal mapping problem with a mixed integer programming (MIP) approach [26]. Another category is to describe the scheduling problem with multiple objective functions and apply various optimizing algorithms to find the optimal solution. Tao *et al.* proposed a case library based hybrid GA to find the Pareto solution which included a multi-parent crossover operator, a two-stage algorithm structure and a case library [12]. Farahnakian *et al.* used an ACO system to solve the VM consolidation problem in the green cloud computing environment [27]. Game theory is also widely used to solve resource allocation problems. Wei *et al.* demonstrated that Nash equilibrium always exists if the resource allocation game has feasible solutions [28]. It has been proved that NSGA-II can attain better spread of solutions and converge better in solving MOO problems [29]. Based on their work, an improved NSGA-II, which include an elitist set and a *k*-means based select strategy, is proposed and evaluated in this paper.

3 KNOWLEDGE-BASED TASK INTERPRETING

3.1 A General Process of Resource Allocation

As discussed above, a distributed and interactive system can facilitate the development of multi-tenant simulation which is performed by various tenants with different requirements within a cloud simulation platform. Each tenant contains various users that share one database instance and application environment to complete the same task. To meet the multi-tenancy need, the system offers a framework for the requirement of negotiating, monitoring and scheduling services based on the Service Level Agreement (SLA) [6].

A general process of resource allocation is shown in Fig. 2, which is based on a knowledge model and an optimization model.

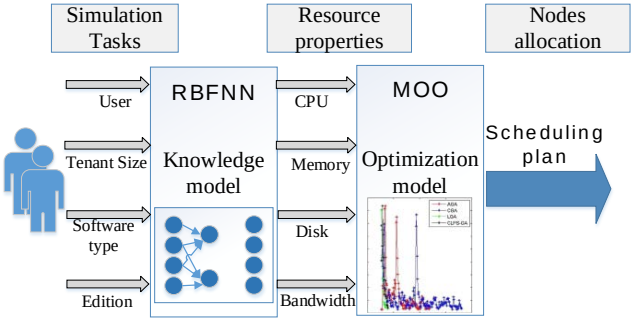


Fig. 2. A general process of resource allocation

a) Knowledge model
In a cloud computing platform, the development process of complex products is decomposed into independent subtask sets that require the collaboration of different developers. Generally, the types and qualities of the resources allocated to subtasks are determined by the developers' experience which is referred to as implicit domain knowledge and is often obtained from previous projects. However, this kind of knowledge is hard to codify for effective and efficient reuse. In order to meet the dynamic demand of services, a RBFNN model is developed for the representation of task-interpreting knowledge, which transforms abstract tenant requirements into specific resource attributes.

b) Optimization model
To allocate resources effectively in a highly heterogeneous cloud environment, a multi-objective optimization model is established by considering the tenant priority, load balance and energy consumption criteria.

To integrate this framework with cloud simulation platforms, all the models and algorithms are developed and encapsulated into a middleware solution which facilitates modularity and expandability of the whole software system.

3.2 A Knowledge Model Based on the RBFNN

According to the application details of cloud simulation platforms, three types of tasks are addressed in this paper.

1) I/O bounded tasks which incorporate a great quantity of data-intensive interactions. A typical example is a combat simulation system. In this case, the overall speed of computation is limited by the data bandwidth between CPU and memory.

2) CPU bounded tasks which are very common in the processing of parallel computing tasks or in the large-scale scientific calculation domain. A typical example is solving the dynamic equations of aircrafts.

3) Memory bounded tasks which often involve various engineering databases and model files and will be better allocated to the nodes with large memory size.

The relations between task types and resource attrib-

utes are so substantially nonlinear that general mathematical models cannot satisfy the requirements [10]. In this paper, a radial basis function neural network (RFBNN) is developed to describe these relationships and address the various complex factors.

The structure of the network is shown in Fig.3, which is in essence a four-layer feedforward network and can be used to effectively identify nonlinear models. The signal propagation mechanism and the basic function of each layer are introduced below.

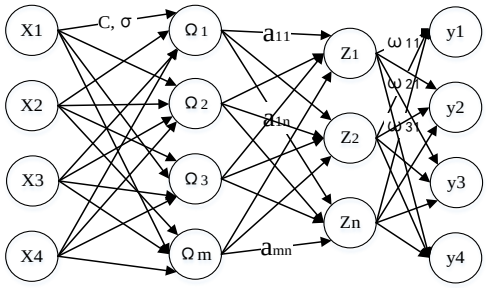


Fig. 3. Structure of the RBFNN

Input layer: each node of this layer corresponds to one input variable. To predict the resource requirements of each task, the input vectors are constructed based on four variables including the user ID, tenant size, software type and software edition. A user generally accomplish some certain types of tasks, for example, developers in charge of system overall design need more bandwidth because of the interactions with engineers from different domains. Thus, user ID and software type can be of advantage to predict the possible behaviors of users and provide a more realistic resource demand plan. Tenant size reflects the size of the problem to be solved and is in proportion to the quantity of resources. Software with a higher edition often needs to be equipped with resources with better qualities. Table 1 gives some examples of the input data. The input nodes in layer 1 transmit input signals to the next layer.

TABLE 1
Some Examples of Input Data

User ID	Tenant Size	Software Type	Software Edition
1	30	Cosim	Standard
2	5	Matlab	Enterprise
3	20	Oracle	Professional

Hidden layer: Each hidden layer node represents a bell shaped radial basis function that is centered on a vector in the feature space. Through linear combination of nonlinear basis function in this layer, the RBFNN can attain a good performance in approximating nonlinear relationships. The mostly used nonlinear functions include the multi-quadratic function, spline function, Gaussian basis function and so on. Gaussian basis function is applied in this paper to realize the nonlinear fitting process, which is shown in Equation (1).

$$\Omega m = \exp(-\frac{\|X-c_m\|^2}{2\sigma_m^2}) \quad (1)$$

In this equation, $\|X - c_m\|^2$ is the square of the distance between the input feature vector X and the center vector c_m for that particular radial basis function and $\{\Omega m\}$ are the outputs from the radial basis functions.

Rule layer: Each node of this layer is a rule node that represents one applicable degree. The outputs from the hidden layer nodes are calculated by the weights on the lines and the weighted sum is computed at the *i*-th output node using Equation (2) where a_{ij} means the weight of the links from the hidden layer to the output layer and Ωj is the value of the hidden layer.

$$Z_i = \sum_{j=1}^{m} a_{ij}\, \Omega j \quad (2)$$

Output layer: the links of output layer will be adjusted in response to various control circumstances. ω_k represents the output action of the k-th rule. The output vector representing the resource attributes is listed in Table 2, which contains CPU size, disk memory size, bandwidth and response time.

TABLE 2
Output of RBFNN

CPU(GB)	Disk(GB)	Bandwidth(Mb/s)	resT(ms)
1.2	150	2.30	3.50
2	260	2.00	3.29
2	240	4.80	5.38

The formula for calculating the numeric output of the model, based on the activation levels of the rules, is given in (3).

$$\mathrm{y(X)} = \sum_{i=1}^{n}\sum_{j=1}^{m} \omega_i \left(a_{ij} \exp\left(-\frac{\|X-c_j\|^2}{2\sigma_j^2}\right) + b_i\right) \quad (3)$$

As can be seen in the above equation, parameter learning of RBFNN contains two parts. The first involves parameters in the hidden layer including the center vector c and normalization vector σ, and the other involves the weight parameters including the weight vector a in the rule layer and ω in the output layer. The location of any RBF in the input space is uniquely specified by its center and width (spread). The output of RBFNN is a weighted sum of the activation levels of the individual RBFs. As shown in Fig.4, the learning algorithm of RBFNN is used to adjust the weights of the links from the hidden layer to the output layer.

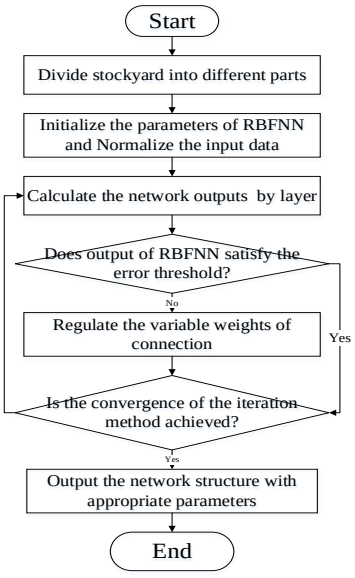


Fig. 4. Flow chart of the RBFNN learning process

The locations of the hidden units and weighted vectors of RBFNN are obtained via the offline methods with the training data collected from previous projects. A data pre-processing procedure is needed before the NN training process. The function of this data pre-processing module is to identify and process abnormal data from the samples as well as to quantify the criteria of resource properties. All the preceding four attribute values of input variables and the predictive goal values are normalized into values with the range of 0 to 1. After the network weight vectors are initialized, the output of each layer is then figured out using (2) and (3). The weighting factors are adjusted based on the deviation between the output value and the expected value. In order to ensure convergence and stability in predicting resource attributes, the network parameters are optimized by repeated training and studying of the samples. The squared sum of maximum error is less than 10^{-6} at the end of each time of iteration.

4 Formulation of the Optimization Model

In this section, the formulation of the resource scheduling problem is detailed with a focus on its main objective functions. With reference to Fig. 1, resource usage of each node is monitored in the middleware layer, while tenant request for resources can be parsed from the tasks submitted in UI layer. The scheduling objectives are obtained in the mathematical optimization model, including maximizing the satisfaction of tenants, minimizing the total cost of service providers and maximizing the load balance of the cloud simulation platform concerned.

4.1 General Definition

4.1.1 Resource Information

$P = \{N_1, N_2, \cdots, N_n\}$ represents a set of nodes with n representing the total number of available nodes. The attributes of each node are as follows:

- CPU Capacity$(capR_i,\ i = 1,2,\cdots n)$: it is defined as the ability and speed of a processor and how many operations it can carry out in a given amount of time, which is typically referred to using the units of Megahertz (MHz) or Gigahertz (GHz).
- Memory Size$(memR_i,\ i = 1,2,\cdots n)$: it is defined as the real size of memory allocated to the node with the units of Gigabytes (GB) or Terabytes (TB).
- Bandwidth$(banR_i,\ i = 1,2,\cdots n)$: it is defined as the rate of data transfer or throughput, measured in bits per second (bit/s). This factor can reflect the speed of interactions among tenants.
- Initiation Time$(iniT_i,\ i = 1,2,\cdots n)$: it is defined as the time taken to initialize a service, which includes VM initiation time and application installation time.
- Price of Node $(costU_i,\ i = 1,2,\cdots n)$: it includes the power cost, software cost and equipment cost. The higher the node price, the higher the quality of the resource concerned.
- Expected value of Delay Time$(delayT_i,\ i = 1,2,\cdots n)$: it is the average value of the node's delay time, which generally has a normal distribution.

4.1.2 Tenant Request Information

$R = \{R_1, R_2, \cdots, R_m\}$ represents a set of requests. In addition to $\{cap_{Tj}, memT_j, banT_j,\ j = 1,2,\cdots m\}$, a request also has the following attributes:

- Level of Request$(level_j,\ j = 1,2,\cdots m)$: it refers to the priority of a tenant, which is defined in the user database.
- Response Time$(resT_j,\ j = 1,2,\cdots m)$: it represents the acceptable time taken by the provider to process a particular customer request.
- Dalay Penalty$(costP_j,\ j = 1,2,\cdots m)$: when the provider's response time is longer than what was specified in the SLA by the tenant, a violation occurs and the provider needs to pay a penalty calculated in (4).

$$costP_j = \begin{cases} a_1, & 0 < delayT_j < t_1 \\ a_1 + \varepsilon_j \times delayT_j, & t_1 < delayT_j \\ 0, & delayT_j < 0 \end{cases} \quad (4)$$

Specifically, $vioT_j = iniT_j^i + delayT_j^i - resT_j$ represents the vilation time; ε_j is related to the priority of tenants which is specified as {1.0, 1.3, 1.5, 1.7, 2.0}.

4.2 Objective Functions

As mentioned earlier in the introduction section, the three main objectives for resource scheduling for the cloud computing environment include maximizing both tenant satisfaction and load balance and minimizing energy consumption reflected in the total cost.

4.2.1 Tenant Satisfaction Factor

Let $\{Tenant_1, Tenant_2, \cdots Tenant_m\}$ represent the set of tenant satisfaction values in the waiting queue and m is the total number of the set. pri_j is a weight factor related to tenants' different priorities, and its value is specified as {1.0, 1.3, 1.5, 1.7, 2.0}. The Satisfaction Factor is calculated as follows:

$$Sat_j = \sum_{j=1}^{m} \frac{1}{m} \cdot pri_j \cdot Tenant_j \quad (5)$$

As different QoS types have different influences on a

tenant's satisfaction, each type is thus given a weight denoted by ω_{ji} and the whole set is normalized as follows,

$$Tenant_j = \sum_{i=1}^{4} \omega_{ji} \cdot \frac{Qua_i - Qua_j}{Qua_j}$$
$$Qua = \{cap, mem, ban, resT\} \quad (6)$$

In Equation (6), Qua_i represents the resource quality of the node allocated to tenant j and Qua_j is the request quality specified in SLA.

4.2.2 Total Cost

The energy efficiency is defined as the amount of energy used by all the pieces of hardware during a period of fulfilling a user request. In this paper, it is calculated by the hardware and software infrastructure cost of the computing resources in the cloud environment, which contains two parts: (1) the hardware and software infrastructure cost of the computing resources in the cloud environment; (2) and the payment for SLA violation.

$$\text{Cost}_j = \sum_{j=1}^{m}(costU_j + costP_j) \quad (7)$$

4.2.3 Load Balancing

The load balancing factor is also composed of two parts, namely balance of load distribution among all the nodes and inequality between different resources in the same node. The former is used to prevent any single node from overloading while the latter ensures that servers are efficiently utilized. If the memory of a node is exhausted, no more tenants can be allocated to this server, regardless of the remaining CPU or other capacities.

Load balancing of all tasks is calculated using (8),

$$loadA = \sigma(cap) + \sigma(mem) + \sigma(sto) \quad (8)$$

$$\sigma(cap) = \sqrt{1/n \cdot \sum_{i=1}^{n}(capT_i/capR_i - \overline{capT/capR})^2} \quad (9)$$

In (9), $capT_i/capR_i$ is the CPU usage of node i; $\overline{capT/capR}$ denotes the average usage of CPU, whose value can be obtained by calculating the mean of elements in the tasks set. Inequality of resource usage for a single node is calculated using (10).

$$loadS = 1/n \cdot \sum_{i=1}^{n}(max\left(\frac{capT_i}{capR_i}, \frac{memT_i}{memR_i}, \frac{banT_i}{banR_i}\right) - min(\frac{capT_i}{capR_i}, \frac{memT_i}{memR_i}, \frac{banT_i}{banR_i})) \quad (10)$$

4.3 Mathematical Formulation

Assume the cloud service provider has m available nodes and n is the number of tasks submitted by its tenants. Each task $T_i = \{capT_i, memT_i, bandT_i, resT_i, level_i, delayC_i\}$ consists of six performance indicators to satisfy. A solution of the scheduling problem is a non-negative matrix of n rows and six columns. Each row of the solution R_i is represented as $\{capR_i, memR_i, bandR_i, iniT_i, costU_i, delayT_i\}$. Based on Equations (4)–(10), the formulas of the three objectives can be rewritten as:

$$\text{Max} \quad \sum_{j=1}^{m} \frac{1}{m} \cdot pri_j \cdot (\sum_{i=1}^{4} \omega_{ji} \frac{Qua_i - Qua_j}{Qua_j}) \quad (11)$$

$$\text{Min} \quad \sum_{j=1}^{m}(costU_j + costP_j) \quad (12)$$

$$\text{Min} \sum \sqrt{1/n \cdot \sum_{i=1}^{n}(capT_i/capR_i - \overline{capT/capR})^2} + 1/n \cdot \sum_{i=1}^{n}(\max(Qua_i) - \min(Qua_i)) \quad (13)$$

The set of constraint functions are defined in (14):

$$capT_i \le capR_i$$
$$memT_i \le memR_i$$
$$bandT_i \le bandR_i \quad (14)$$

5 The Improved Multi-Objective Genetic Algorithm

The pseudo code of the improved multi-objective genetic algorithm is shown in Algorithm 1. Description of the procedure is given as follows. In the initialization phase, a set of parameters is initialized, including the population, the elitist archive and the mutation probability for solution updating. At the beginning of each time of iteration, the non-dominated solutions are found and sorted according to the *k*-means calculation result. The selection, crossover and mutation operations are then performed in order. After all the populations finish constructing their solutions, a global update is performed on each solution of the current Pareto set. This procedure is repeated until the maximum number of iteration (IterMax) is reached and the solutions in ES are the obtained Pareto-optimal solutions.

Algorithm 1. Main body of the improved KGA

Input: Set of available nodes with their resource utilization and set of tasks with the resource demand, Set of parameters (*PopSize, IterMax, EleSize,*ρ_1, ρ_2,)

Output: Pareto-optimal solutions (ES)

1. FS ⇐ find the basic feasible solution from all the available nodes
2. PS ⇐ initialize the population with *PopSize* randomly generated solutions from FS
3. PS_N ⇐ calculate the fitness value and find the non-dominated solutions in PS
4. ES ⇐ initialize the elitist archive by PS_N
5. **for** *iter* = 1 to *IterMax* **do**
6. **for** each S ∈ PS_N **do**
7. *pros* ⇐ calculate the clustering results, select probability and sort
8. **end for**
9. FAS1 ⇐ select $\rho_1 \times PopSize$ solutions from PS_N based on *pros*
10. FAS2 ⇐ select $\rho_1 \times PopSize$ solutions from PS_N based on *pros*
11. CroS ⇐ perform single point crossover by randomly selecting the crossover point of solution from FAS1 and FAS2
12. MutS ⇐ mutate new offspring with a mutation probability of ρ_2 using polynomial mutation
13. PS_{new} ⇐ update the solution with FAS1 ∪ FAS2 ∪ CroS ∪ MutS
14. ES ⇐ update the elitist archive according to PS_{new}
15. **end for**

To find good and diverse Pareto-optimal solutions, an

elitist preserving strategy is developed by maintaining a fixed-size archive.

5.1 Selection Strategy Based on the *K*-means Approach and Proportional Distribution

A selection strategy based on the *k*-means algorithm and proportional distribution is used to choose the best solutions from the archive in order to spread the particles along the Pareto front.

Algorithm 2. The select operator
Input: Set of initialized population, Set of parameters (*SelectSize,* PS)
Output: Selected solutions (FAS)

1. $PS_N \Leftarrow$ find the non-dominated solutions in PS
2. **for** each $S \in PS_N$ **do**
3. *KNum* $\Leftarrow$ calculate the clustering results according to (15)
4. **end for**
5. *pro* $\Leftarrow$ calculate the selected probability and sort according to (16)
6. *ms* $\Leftarrow$ generate *SelectSize* random number and sort
7. *fitin = 1, newin = 1*
8. **while** *newin* ≤ *selectSizse* **do**
9. **if** *ms(newin) < pro(fitin)* **then**
10. FAS(*newin*) $\Leftarrow$ update with PS_N(*fitin*), update newin
11. **else then**
12. update newin
13. **end if**
14. **end while**

$$KNum_i = \sum_{S \in S_N} sgn\left(r - \sqrt{\sum_{k=1}^{3}\left[\frac{f_k(S)-f_k(S_i)}{{f_k}^{max}(S_N)-{f_k}^{min}(S_N)}\right]^2}\right) \quad (15)$$

$$pro_i = \frac{1/KNum_i}{\sum_{i=1}^{K} 1/KNum_i} \quad (16)$$

In Equations (15) and (16), $f_j(S_i)$ is the objective value of solution S_i S_i; ${f_k}^{max}(S_N)$ and ${f_k}^{min}(S_N)$ are the maximum and minimum value of the kth objective function, respectively; sgn is the sign function which means that $KNum_i$ gets larger as the unit sphere contains more non-dominated solutions.

5.2 Performance Indicators

Generally, the performance of different MOO algorithms is measured by the quality of the Pareto-optimal sets [32]. Based on a comparative study of different models, the three indicators below are chosen and elaborated in this research.

The first is the spacing (S) indicator, which measures the distribution of a Pareto-optimal solution set. A more uniformly distributed solution can provide users with more options to find representative and satisfactory solutions, which can be calculated as follows:

$$S(S^*) = \sum_{S_i \in S^*} sgn(\bar{d} - d_i) \quad (17)$$

In Equation (17), d_i denotes the squared Euclidean distance between solution S_i and the nearest optimal solution; $\bar{d}$ is the mean value of d_i; and sgn is the sign function.

The second is the convergence indicator, which can be understood as the average distance of the obtained Pareto-optimal solution (S^*) from the true Pareto-optimal set ($\overline{S^*}$). In mathematical forms, it can be calculated as follows:

$$\text{convergence}(S^*) = \sqrt{\sum_{i=1}^{|S^*|} d_i^2 \Big/ |S^*|} \quad (18)$$

$$d_i = min_{j=1}^{|\overline{S^*}|} \sqrt{\sum_{k=1}^{3}\left[\frac{f_k(S^*)-f_k(\overline{S^*})}{f_k^{max}-f_k^{min}}\right]^2} \quad (19)$$

In Equation (18),$|S^*|$, $|\overline{S^*}|$ indicates the number of solutions in the set S^*, $\overline{S^*}$; d_i is the normalized Euclidean distance between the ith solution (S^*) and the true Pareto-optimal set; and f_k^{max} and f_k^{min} are the maximum and minimum values of the kth objective function in the true Pareto-optimal set, respectively.

The third is the maximum spread (MS) indicator. It measures the distribution of a solution set (S^*) by calculating the normalized distance between the boundary solutions as follows:

$$MS(S^*) = \sqrt{\frac{1}{3}\sum_{k=1}^{3}\left[\frac{\max_{s \in S^*} f_k(S) - \min_{s \in S^*} f_k(S)}{f_k^{max}-f_k^{min}}\right]^2} \quad (20)$$

6 Experiments and Performance Evaluation

6.1 Simulation Task

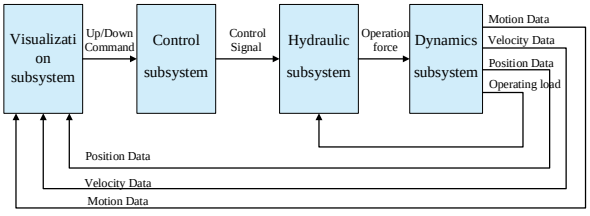


Fig. 5. Block diagram of the simulation case

COSIM-CSP is a collaborative simulation framework that provides a platform on which developers can perform multidisciplinary collaborative simulation of virtual prototyping and large scale co-simulation [31]. The case study used in this research is the collaborative simulation problem of virtual-prototyping supported aircraft landing gear design on the COSIM-CSP platform. In this case, design departments in charge of different domain modeling and simulation tasks correspond to different tenants while various design engineers are given different user identities. As shown in Fig. 5, the simulation system is mainly composed of four subsystems: control subsystem, hydraulic subsystem, multi-body dynamics subsystem, and visualization subsystem. The simulation problem involves many disciplines such as the control discipline, mechanical discipline, top-level design, etc. In the course of simulation advancement, the control subsystem starts the control system simulation with gesture information from the platform, and then sent the result (control force) back to the platform. The hydraulic subsystem receives the control signal and solves the hydraulic equations. Table 3 enumerates the requirements of different simulation tasks in the collaborative simulation. The SLA-based scheduling algorithm is integrated into the COSIM-CSP platform and is proved to be able to support SLA of tenants with different levels of priority effectively.

TABLE 3
Some Examples of Input Data

Task	ID	Tenant	Software	Edition
Visualization	1	5	Catia	Standard
Control	2	10	Matlab	Enterprise
Hydraulic	3	5	Fluent	Enterprise
Dynamics	4	10	Adams	Enterprise
Overall design	**5**	5	Cosim	Standard

6.2 Task Interpreting

The performance of the proposed model for estimating the required resources for each tenant is compared with the performance of a grey exponent static model.

Table 4 shows the mapping between the task sample and the variables. Specifically, $\oplus_{11}$ denotes the low version of software I while $\oplus_{13}$ denotes the high version of software I.

TABLE 4
Training Data for Task Parsing

Serial Number of Data		1	2	3	4	5	6
$\oplus_{11}$	δ_{11}	1	1	1	0	0	0
$\oplus_{12}$	δ_{12}	0	0	0	1	1	0
$\oplus_{13}$	δ_{13}	0	0	0	0	0	1
$\oplus_{21}$	δ_{21}	0	0	1	0	1	0
COM_g	45.82	14.69	24.49	7.75	14.38	20.45	–

According to the GSEM $_{(1,\ 2)}$ model, the memory size can be formulated by using Equation (21).

$$\mathrm{Mem}S_g = \exp\left(a_{11}COM_g^r + b_{12}\delta_{12} + b_{13}\delta_{13} + b_{21}\delta_{21}\right) \quad (21)$$

The least squares approach is applied to solve the nonlinear equations progressively by using a series of linear equations. Coefficients of the equation are obtained as follows,

$$\gamma = 0.215$$

$$[a_{11}\ b_{11}\ \ b_{12}\ b_{13}\ b_{21}\ ]^T = [1.508\ \ 0.698\ \ 1.120\ \ -0.357]^T$$

The two models are applied to a set of experiments for testing the memory sizes required by different tenants when the size of test data varies. The mean values of the prediction errors are shown in Table 5.

TABLE 5
Prediction errors of GSEM and RBFNN

Training data	5	10	15	20	25	30
GSEM(1,2)	21.5	10.2	12.1	14.03	19.2	25.4
RBFNN	35.6	23.4	14.6	11.3	10.9	9.6

Table 5 shows that the grey model forecast the output effectively when the training data are not many while RFBNN has a good learning ability and can easily express quantitative knowledge. Fig. 6 shows the variation trend of prediction error in response to size changes of training data. Fig. 7 shows the evolution curve of the RBFNN.

Table 6 gives detailed information about the required resources according to the RBFNN model.

Fig. 6. Prediction error of GSEM and RFBNN

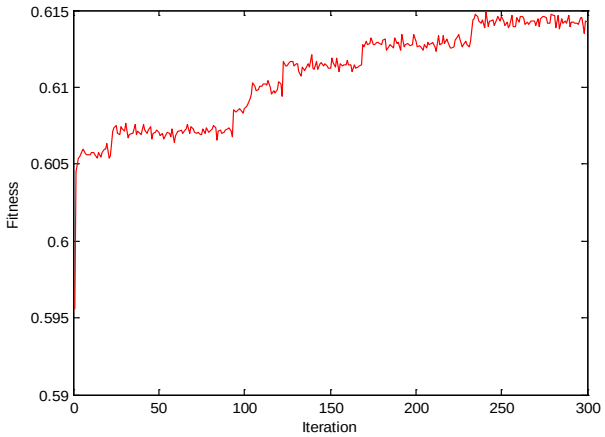


Fig. 7. Evolution curve of the RBFNN

TABLE 6
Resources Requirements of Tasks

ID	CPU(GB)	Disk(GB)	Bandwidth(Mb/s)	resT(ms)	level
1	1.2	200	2.00	3.50	3.00
2	2	200	2.00	3.31	1.00
3	2	240	4.00	5.01	2.00
4	2.4	160	3.00	5.59	2.00
5	3	260	3.00	5.60	1.00

6.3 Resource Scheduling

The COSIM-CSP cloud computing platform is equipped with 200 nodes, which have different resource attributes. At the beginning of the simulation, it is assumed that all nodes are available and the resources of each node are undividable. In general, the nodes with higher CPU frequency and larger memory size consume more energy and energy consumption is directly related to total cost of a task in the proposed model. Another factor influencing total cost is the maintenance status. The simulation task mentioned above can be divided into nine subtasks, two of which have a higher priority than others. Though different tasks are divided into the CPU-bounded ones, the I/O bounded ones and the memory-bounded ones, each specific one is defined by specifying explicit numeric values according to the parsing result of the RBFNN. Task requirements are represented in the form of a non-negative matrix with 9 rows (each one corresponds to a task) and 6 columns (each one corresponds to a resource demand). The available resources are represented in the

form of a non-negative matrix with 200 rows (each one corresponds to a computing node) and 6 columns (each one corresponds to a specific value of resource capacity).

Numerical experiments are conducted using Matlab R2011a on a PC with a Core i7 3.40 GHz CPU. As the performance of an evolution algorithm is influenced by its initial setting of parameters such as *Population Size, Crossing Rate, Mutation Rate, and the maximum number of iteration*, the best parameters values of the basic GA are adopted. A parameter *EleSize* which is specific to KGA is set to different values in order to investigate its influence on the performance of the algorithm.

The testing experiments and their results are summarized as follows. First, the improved algorithm is applied to the collaborative simulation task to test its effectiveness. As mentioned above, the performance of a multi-objective optimization algorithm can be determined by the Pareto solutions generated. Secondly, the performances of KGA, NSGA-II and PSO are compared by calculating the mean values of different objectives in the elitist set and the quality of the Pareto-optimal sets.

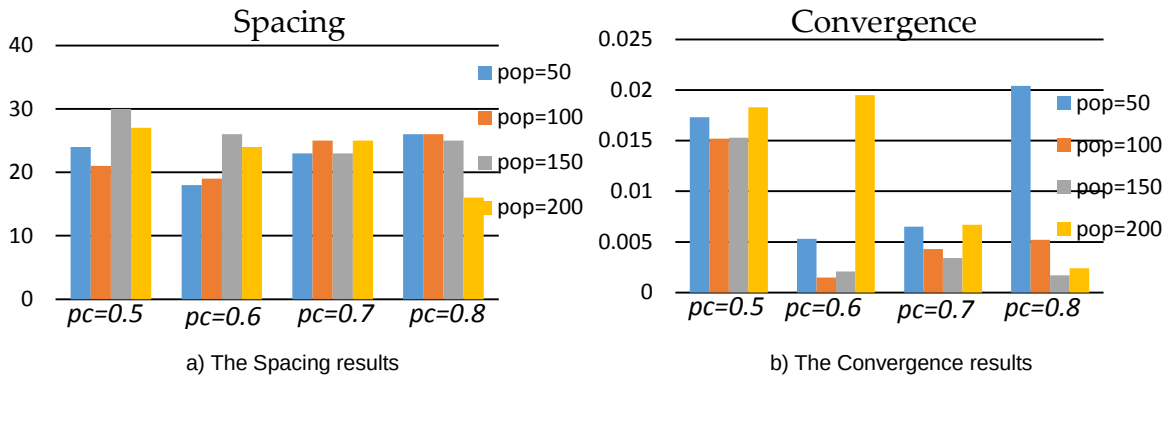
6.3.1 Parameter setting

To investigate the influence of different parameters on the performance of the proposed algorithm, each parameter is set in four levels. Obviously, a huge amount of computational efforts have to be taken if the algorithm is tested under all possible combinations. To overcome this drawback, the Taguchi method of design of experiment (DOE) is applied in this study, which uses orthogonal arrays to decrease the number of experiments. According to the number of parameters and the number of factor levels, the orthogonal array $L_{16}(4^3)$ in Table 7 is selected for conducting the experiments.

For each parameter combination, the average results for the three indicators are listed in Table 7. Accordingly, Fig. 8 shows the factor level trend regarding different indicators. Since the solution set with a higher diversity value, a lower convergence value and a higher spread value is preferred, a good choice of parameter combination is suggested as $pop = 100, pc = 0.8\ and\ pm = 0.1$.

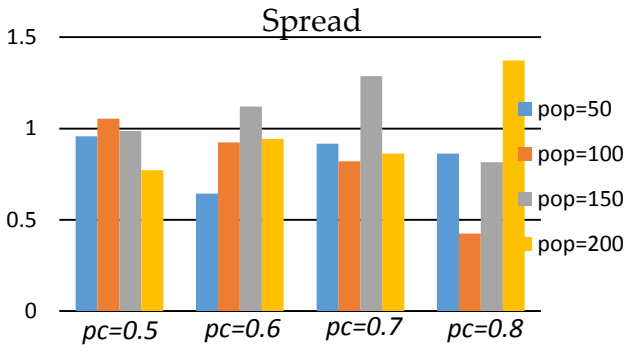
Table 7
The Orthogonal Array and Experimental Results

Test number	Factor level			Average results		
	pop	*pc*	*pm*	Spacing	Convergence	Spread
1	50	0.5	0.05	24	0.0173	0.9563
2	50	0.6	0.1	18	0.0053	0.6435
3	50	0.7	0.15	23	0.0065	0.9164
4	50	0.8	0.2	26	0.0204	0.8634
5	100	0.5	0.1	21	0.0152	1.0536
6	100	0.6	0.05	19	0.0015	0.9235
7	100	0.7	0.2	25	0.0043	0.8213
8	100	0.8	0.15	26	0.0052	0.4243
9	150	0.5	0.15	30	0.0153	0.9875
10	150	0.6	0.2	26	0.0021	1.1205
11	150	0.7	0.05	23	0.0034	1.2865
12	150	0.8	0.1	25	0.0017	0.8145
13	200	0.5	0.2	27	0.0183	0.7706
14	200	0.6	0.15	24	0.0195	0.9425
15	200	0.7	0.1	25	0.0067	0.8634
16	200	0.8	0.05	16	0.0024	1.3721



a) The Spacing results

b) The Convergence results



c) The Spread results

Fig. 8. The factor level results regarding different indicators

6.3.2 *Comparison between KGA and other MOO algorithms*

Two multi-objective optimization algorithms, namely NSGA-II and MOPSO, are chosen to be compared with the proposed algorithm. The initial settings of these algorithms are given in Table 8. These parameters remain unchanged throughout all the experiments.

In Fig.9, a three-dimensional plot is used to display the distribution of solutions in the elitist set with its three axes corresponding to the total cost, satisfaction and load balancing values, respectively. To provide more details regarding the solutions obtained, three additional two-dimensional scatter diagrams are also included in Fig. 9 to show the distribution of solutions when any two objectives are considered. For example, in the second figure, the vertical axis is used to show satisfaction values while the horizontal one is used to show total cost. It can be seen from the plots that the output solutions contain a set of points located on the Pareto front.

Table 8
Parameter Setting

Algorithms	Parameters	Settings
KGA	Population size (*pop*), Crossover probability (*pc*), Mutation probability (*pm*), elitist archive size (*ele*)	100, 0.8, 0.1, 50
NSGA-II	Population size (*pop*), Crossover probability (*pc*), Mutation probability (*pm*)	100, 0.8, 0.1
MOPSO	Mutation probability (*pm*), Cognitive crossover probability (pc_1), Social crossover probability (pc_2)	0.4, 0.7, 0.7

TABLE 9
Performance Indicators Results of the Test Experiments

Task number	Iteration	Spacing			Convergence			Spread		
		KGA	MOPSO	NSGA-II	KGA	MOPSO	NSGA-II	KGA	MOPSO	NSGA-II
5	100	24	21	25	0.0152	0.0554	0.0268	1.0234	0.4548	0.3459

	200	26	23	26	0.0124	0.0425	0.0197	1.2457	0.5614	0.8651
	300	28	25	26	0.0087	0.0378	0.0164	1.5154	0.8749	1.0248
	500	30	25	27	0.0024	0.0405	0.0162	1.4718	0.9146	1.1341
Average		27	24	26	0.0097	0.0441	0.0198	1.3141	0.7014	0.8425
10	100	24	17	21	0.0634	0.1647	0.1257	0.8165	0.2178	0.2865
	200	25	18	23	0.0565	0.1426	0.1164	1.0247	0.4617	0.6451
	300	26	21	25	0.0458	0.1250	0.1028	1.3154	0.7254	1.01258
	500	27	23	25	0.0326	0.1364	0.0862	1.2416	0.8164	1.0047
Average		26	20	24	0.0496	0.1422	0.1078	1.0996	0.5553	0.7372
20	100	21	16	17	0.3653	0.4508	0.3987	0.6879	0.5548	0.6135
	200	22	18	18	0.2465	0.7652	0.3694	0.8415	0.7247	0.7326
	300	24	21	18	0.1798	0.6054	0.2819	0.9220	0.7364	0.8927
	500	24	21	17	0.1247	0.3657	0.2410	0.9152	0.8579	0.9454
Average		23	19	18	0.2291	0.5468	0.3228	0.8417	0.7185	0.7961

TABLE 10
Objective Results of the Test Experiments

Task number	Iteration	Total cost			Satisfaction			Load balance		
		KGA	MOPSO	NSGA-II	KGA	MOPSO	NSGA-II	KGA	MOPSO	NSGA-II
5	100	103.43	102.91	103.24	-33.15	-34.67	-34.25	0.162	0.159	0.166
	200	101.81	101.56	101.68	-33.99	-35.48	-35.62	0.156	0.152	0.154
	300	99.78	100.15	99.35	-34.92	-35.82	-36.09	0.151	0.148	0.147
	500	99.03	99.24	98.76	-36.16	-36.04	-36.11	0.143	0.147	0.145
10	100	106.24	106.59	106.78	-31.84	-32.15	-31.68	0.171	0.176	0.175
	200	104.75	105.94	104.65	-32.45	-33.94	-32.17	0.164	0.161	0.167
	300	101.87	102.28	102.37	-34.03	-34.26	-34.59	0.151	0.158	0.154
	500	100.34	101.76	101.06	-35.72	-35.11	-35.26	0.145	0.15	0.151
20	100	110.48	109.84	111.23	-29.17	-30.19	-30.27	0.192	0.189	0.190
	200	108.67	105.46	109.64	-31.84	-32.65	-31.65	0.175	0.163	0.178
	300	104.19	102.37	107.62	-33.16	-35.71	-34.51	0.159	0.157	0.164
	500	99.75	101.45	100.21	-36.07	-36.63	-35.45	0.150	0.153	0.156

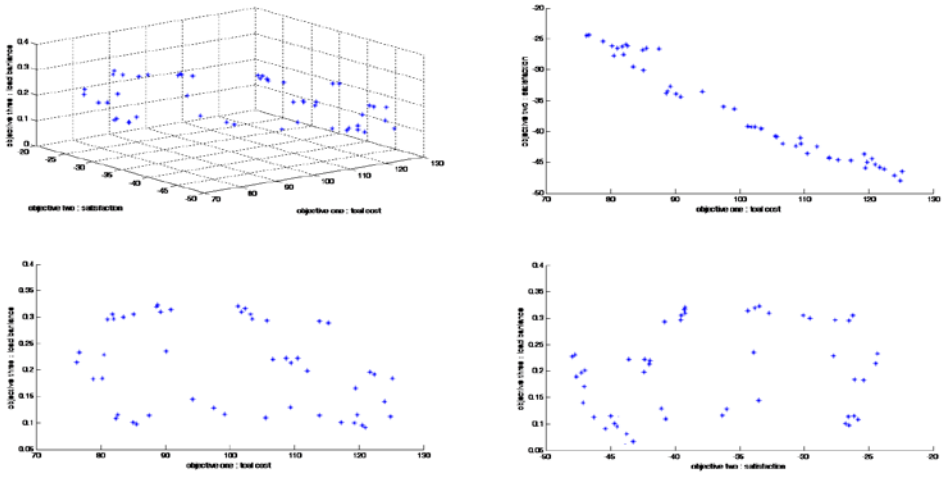


Fig. 9. Distribution of solutions in the elitist set

The computational results are shown in Table 9 and Table 10, including the mean values of the three objectives and the average running time for various numbers of iteration and different task sizes. In terms the running time, it is shown that KGA runs faster than MOPSO and NSGA-II under various conditions, which is mainly ascribed to the difference in computational complexity. When the task size is 5, there is not much difference between the final outcomes of these two optimization algorithms, which can be figured out from the mean of the objective values. As the size of the problem gets larger, KGA performs better than MOPSO and NSGA-II in terms of both the optimization results and performance indicators. However, the objective value of MOPSO is smaller than that of KGA and NSGA-II when the number of iteration is 100 or 200, meaning that in this case MOPSO has a faster convergence speed than KGA.

7 Conclusion and Future Directions

This paper presents an effective solution for resource scheduling in a heterogeneous and collaborative cloud computing environment to meet tenants' diverse simulation requirements. A novel mathematical model is formulated to represent the complex scheduling problem and an improved multi-objective genetic algorithm is proposed based on the elitist archive and the *k*-means approaches. By capturing and reusing engineering experience, a RBFNN model is established to interpret simulation tasks and then match them to resource requirements in terms of their quantities and qualities. The RBFNN model is compared with a grey exponent static model and simulation results show that it has attached a good learning capability. Numerical experiments have also been conducted to compare the performances of the proposed KGA and PSO. Testing results in a case study have demonstrated the good performances of KGA in terms of both convergence and running time for solving collaborative simulation problems.

In our future work, the influence of various parameters in the mathematical model such as the node price and the priority weight will be analyzed in detail. Besides, more practical applications will be considered such as the learning effect and the synchronous task constraints. Furthermore, the algorithms with better performance in terms of efficiency will be developed for relatively large-scale problems with complex constraints.

Acknowledgement

This research is supported by the National Natural Science Foundation of China （Grant No. 61374163）, the National Key Technology R&D Program (Grant No. 2012BAF15G00), and the National High Technology Research and Development Program (863 Program) of China (Grant No. 2013AA041302). The original version of this paper was presented at the 19th IEEE CSCWD Conference held in Calabria, Italy in May 2015.

References

[1] X. Xu, "From Cloud Computing to Cloud Manufacturing, "*Robotics and Computer-integrated Manufacturing*, vol. 28, no.1, pp. 75-86, Feb. 2012.

[2] L. Wu, S. K. Garg, S. Versteeg, and R. Buyya, "SLA-Based Resource Provisioning for Hosted Software-as-a-Service Applications in Cloud Computing Environments," *IEEE Trans. Services Computing*, vol.7, no.3, pp. 465-485, Sep. 2014, doi: 10.1109/TSC.2013.49.

[3] J. Espadas, A. Molina, G. Jiménez, et al, "A Tenant-Based Resource Allocation Model for Scaling Software-As-A-Service Applications over Cloud Computing Infrastructures," *Fut. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 273-286, Jan 2013.

[4] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems,'' *Fut. Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599-616, June 2009.

[5] Boeing Edge, Referenced on May 10 2015. [Online]. Available: http://www.boeing.com/commercial/boeing-edge/

[6] Y. Y. Ran, J. Yang, S. B. Zhang, and H. S. Xi, "SLA-Driven Dynamic Resource Provisioning for Service Provider in Cloud Computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, 2013, pp. 408-413, doi: 10.1109/GLOCOMW.2013.6825022.

[7] L. Wu, S. K. Garg, and R. Buyya, "SLA-Based Admission Control for a Software-As-A-Service Provider in Cloud Computing Environments," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1280-1299, Sept. 2012.

[8] L. Wu, S. K. Garg, and R. Buyya, "Sla-Based Resource Allocation for Software as a Service Provider (Saas) in Cloud Computing Environments," in *Proc. 11th IEEE/ACM Int'l Symp. CCGrid*, Los Angeles, CA, USA, 2011, pp. 195-204.

[9] R. Buyya, S. K. Garg, and R. N. Calheiros, "SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions," in *Proc. 2011 International Conference on Cloud and Service Computing (CSC)*, Hong Kong, 2011, pp. 1-10.

[10] D. Krajnc and P. Glavič, "A Model for Integrated Assessment of Sustainable Development," *Resources, Conservation and Recycling*, vol. 43, no. 2, pp. 189-208, Jan 2005.

[11] N.M.P. Bocken, S.W. Short, P. Rana, and S. Evans, "A Literature and Practice Review to Develop Sustainable Business Model Archetypes," *Journal of Cleaner Production*, vol. 65, pp. 42-56, Feb 2014.

[12] F. Tao, Y. Feng, L. Zhang L, and T. W. Liao. "CLPS-GA: A Case Library and Pareto Solution-Based Hybrid Genetic Algorithm for Energy-Aware Cloud Service Scheduling," *Applied Soft Computing*, vol. 19, pp. 264-279, Jun 2014.

[13] J. Yu, R. Buyya, and K. Ramamohanarao. "Workflow Scheduling Algorithms for Grid Computing," *Metaheuristics for scheduling in distributed computing environments*. Springer Berlin Heidelberg, pp. 173-214, 2008.

[14] A. Y. Zomaya and Y. H. Teh, "Observations on Using Genetic Algorithms for Dynamic Load-Balancing," *IEEE Trans. Parallel and Distributed Systems*, vol.12, no. 9, pp. 899-911, Sep 2001.

[15] K. M. Sim, "Agent-Based Cloud Computing," *IEEE Trans. Services Computing*, vol. 5, no. 4, pp. 564-577, Nov 2012.

[16] F. Tao, C. Li, and T. Liao, "BGM-BLA: A New Algorithm for Dynamic Migration of Virtual Machines in Cloud Computing," *IEEE Trans. Services Computing*, Mar 2015, doi: 10.1109/TSC.2015.2416928.

[17] P. C. Church and A. M. Goscinski, "A Survey of Cloud-Based Service Computing Solutions for Mammalian Genomics," *IEEE Trans. Services Computing*, vol. 7, no. 4, pp. 726-740, Dec 2014, doi: 10.1109/TSC.2014.2353645.

[18] J. Yang, J. Qiu, and Y. Li, "A Profile-Based Approach to Just-In-Time Scalability for Cloud Applications," in *Proc. IEEE International Conference on Cloud Computing (CLOUD)*, Bangalore, pp. 9-16, Sept 2009.

[19] H. Wang, A. L. Johnson, and R. H. Bracewell, "The Retrieval of Structured Design Rationale for the Re-Use of Design Knowledge with an Integrated Representation," *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 251-266, Apri 2012.

[20] F. Behloul, BPF. Lelieveldt, and A. Boudraa, "Optimal Design of Radial Basis Function Neural Networks for Fuzzy-Rule Extraction in High Dimensional Data," *Pattern Recognition*, vol. 35, no. 3, pp. 659-675, Mar 2002.

[21] L. Zhang, Y. Luo, F. Tao, et al, "Cloud Manufacturing: A New Manufacturing Paradigm," *Enterprise Information Systems*, vol.

8, no. 2, pp. 167-187, 2014.

[22] X. Lin, Y. Wang, and Q. Xie, "Task Scheduling with Dynamic Voltage and Frequency Scaling for Energy Minimization in The Mobile Cloud Computing Environment," *IEEE Trans. Services Computing*, vol. 8, no. 2, pp. 175-186, Apri 2015, doi: 10.1109/TSC.2014.2381227.

[23] J. M. Alcaraz and A. J. Gutierrez, "Mon-Paas: An Adaptive Monitoring Platform as a Service for Cloud Computing Infrastructures and Services," *IEEE Trans. Services Computing*, vol. 8, no. 1, pp. 65-78, Feb 2015, doi: 10.1109/TSC.2014.2302810.

[24] S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of Re-Source Provisioning Cost in Cloud Computing," *IEEE Trans. Services Computing*, vol. 5, no. 2, pp. 164-177, May 2012, doi: 10.1109/TSC.2011.7.

[25] M. Yu, Y. Yi, J. Rexford, et al, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 2, pp. 17-29, 2008.

[26] C. Papagianni, A. Leivadeas, S. Papavassiliou, et al, "On the Optimal Allocation of Virtual Resources in Cloud Computing Networks," *IEEE Trans. Computers*, vol. 62, no. 6, pp. 1060-1071, Apri 2013, doi: 10.1109/TC.2013.31.

[27] F. Farahnakian, A. Ashraf, T. Pahikkala, et al, "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Trans. Services Computing*, vol. 8, no. 2, pp. 187-198, Apri 2015, doi: 10.1109/TSC.2014.2382555.

[28] G. Wei, A. V. Vasilakos, Y. Zheng, and N.Xiong, "A Game-Theoretic Method of Fair Resource Allocation for Cloud Computing Services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252-269, Nov 2010.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, Aug 2002.

[30] B. Li, X. Chai, L. Zhang, et al, "New Advances of the Research on Cloud Simulation," *Advanced /methods, Techniques, and Applications in Modeling and Simulation*. Springer Japan, pp. 144-163, 2012.

[31] A. Andrieux, K. Czajkowski, A. Dan, et al, "Web Services Agreement Specification (WS-Agreement)," *Open Grid Forum*, September 2005.

[32] J. Dong, X. Zhang, and T. Xiao, "A Hybrid PSO/SA Algorithm for Bi-Criteria Stochastic Line Balancing with Flexible Task Times and Zoning Constraints," *Journal of Intelligent Manufacturing*, pp. 1-15, Jul 2015.

[33] Y. Gao, H. Guan, Z. Qi, et al, "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230-1242, Dec 2013.

[34] R. Raju, J. Amudhavel, N. Kannan, et al, "A Bio Inspired Energy-Aware Multi Objective Chiropteran Algorithm (EAMOCA) for Hybrid Cloud Computing Environment," in *Proc. IEEE International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Coimbatore, pp. 1-5, 2014.

[35] F. Farahnakian, T. Pahikkala, P. Liljeberg, "Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing," in *Proc. IEEE 8th International Conference on Cloud Computing (CLOUD)*, New York, pp. 381-388, 2015.

[36] C. J. Chen, J. Zhang, J. Li, X. Li, "Resource Virtualization Methodology for On-Demand Allocation in Cloud Computing Systems," *Service Oriented Computing and Applications*, vol. 7, no. 2, pp. 77-100, Jun 2013.

[37] L. Ramachandran, N. C. Narendra, K. Ponnalagu, "Dynamic Provisioning in Multi-Tenant Service Clouds," *Service Oriented Computing and Applications*, vol. 6, no. 4, pp. 283-302, Dec 2012.

Gongzhuang Peng is currently a doctoral student at the National Engineering Research Centre of Computer Integrated Manufacturing System (CIMS-ERC) in Tsinghua University, Beijing, China. He received his Bachelor degree in the Department of Automation From Beihang University, Beijing, China in 2012. His research interests are focused on multi-disciplinary modeling, collaborative simulation and knowledge management.

Hongwei Wang is currently a Senior Lecturer in Engineering Design at School of Engineering, University of Portsmouth. Prior to joining the school, he completed his PhD at the Engineering Design Centre of the University of Cambridge. He obtained a BEng (Electronic and Information Sciences) with distinction from Zhejiang University and an MSc (Control Science and Engineering) with distinction from Tsinghua University in 2004 and 2007, respectively. His research interests are in the application of information and computing technologies to address specific design issues, for example, knowledge and information management, collaborative product development, collaborative modeling and simulation and sustainable design.

Jietao Dong is currently a doctoral student at Department of Automation, Tsinghua University, Beijing, China. He received his bachelor degree from Beihang University in 2010. His main research interests include intelligent optimization, production scheduling and simulation. He has got five publications relative to intelligent manufacturing in well-established journals and major conferences.

Heming Zhang is currently a professor at the National Engineering Research Centre of Computer Integrated Manufacturing System (CIMS-ERC) in Tsinghua University, China. He received his PhD degree in Mechanical Engineering from Zhejiang University, China in 1995. He joined the faculty of Department of Automation in Tsinghua University after completing a two-year Postdoctoral Fellowship in CIMS-ERC. His research interests include multidisciplinary modeling and collaborative simulation, service oriented modeling and simulation, concurrent and collaborative design.